Discovery and Reasoning in Mathematics

DISCOVERY AND REASONING IN MATHEMATICS'

Alan Bundy

Department of Artificial Intelligence,
University of Edinburgh

Abstract

We discuss the automation of mathematical reasoning, surveying the abilities displayed by human mathematicians and the computational techniques available for automating these abilities. We argue the importance of the simultaneous study of these techniques, because problems inherent in one technique can often be solved if it is able to interact with others.

Keywords

Reasoning, mathematics, deduction, search, learning, proof analysis.

1. Introduction

A major goal of artificial intelligence is to automate reasoning. Solutions to this goal will have both technological applications, enabling us to build more powerful expert systems, and scientific applications, providing models to compare with human reasoning. Mathematics is an excellent domain for exploring the automation of reasoning because:

(a) it provides a wide range of examples of reasoning, from the simple and shallow to the complex and deep;

(b) it is possible to detach this reasoning from other considerations, such as sensory input of data; and,

(c) to a first approximation, the problems of knowledge representation have been solved.

Compare mathematics with other AI domains in which reasoning plays a part, e.g. natural language understanding or visual perception. In mathematics one is not bogged down with huge amounts of noisy data, nor concerned that what seems to be a difficult reasoning issue may evaporate if the knowledge representation were changed. One is free to concentrate on the reasoning problems and then make an attempt to translate any solution found to other domains.

In this paper I discuss the automation of mathematical reasoning, by which I mean any cognitive activity that mathematicians engage in as mathematicians. The main theme will be that mathematical reasoning consists of more than just theorem proving, and that the simultaneous automation of other reasoning processes, e.g. learning, simplifies the automation task. Each reasoning process outputs knowledge but also demands knowledge as input. If a computational technique for automating a particular reasoning process is studied in isolation then the provision of its input knowledge may prove a major barrier to automation. But the input knowledge of one technique is the output knowledge of another, so that the techniques fit together in an intercommunicating network. Some techniques for automated reasoning may involve search. In isolation the search involved may be computationally expensive. But one technique may be used to control the search of another, especially if the two techniques are co-routined or even merged into one. The power of a system in which the various techniques interact in well-crafted ways will be more than just the sum of the power of the parts.

I advocate the simultaneous study of all the the processes of mathematical reasoning with particular emphasis on the possible interactions between the techniques that automate them.

2. Reasoning Abilities in Mathematics

What processes are involved in mathematical reasoning? In this section we discuss the various processes and the AI techniques that have been proposed to automate them. More details about these techniques can be found in [Chang &, Lee 73. Bundy 83].

The first problem we face is how to classify mathematical reasoning into different processes - what my psychology friends call defining the *task ecology.* Our starting point is a 'folk ecology' of reasoning processes, i.e. the terms used in pre-scientific discussion of reasoning. We will call these *mathematical abilities.* With these abilities we must associate *computational techniques* which implement them. These techniques will form our scientific classification. But the association of abilities to techniques not a neat 1-1 mapping. We are likely to find many techniques to associate with each ability. Some abilities, e.g. learning, have so many diverse techniques associated with them that they seem highly unsuitable as scientific categories, whereas the techniques associated with some abilities, e.g. theorem proving, all have a strong

This research was supported by SERC grant. number GR/C/20826.

family resemblance. Some abilities require a combination of techniques. Consider, for instance, analogy which requires separate techniques for finding and then using the analogy. Each of these techniques will require sub-techniques. Some of these techniques may contribute to several different abilities. Consider, for instances, resolution-type deduction techniques, which are ubiquitous.

We list below the mathematical abilities we will be considering under the heading of mathematical reasoning and mention some of the techniques which have been used to implement these abilities.

- Proving theorems: The ability that has received the most attention in the AI study of mathematical reasoning, almost to the exclusion of all others, has been theorem proving. Automated theorem proving has been an important subfield of AI throughout its history. The main techniques required to automate theorem proving are *deduction* which involves *search control.* Deduction traverses a search space of legal inference steps. Search control decides which of these steps to try.

- Formalizing Problems: However, mathematicians, especially applied mathematicians, spend a large part of their time translating informal problem statements into mathematical formulae to which deduction, etc. may be applied. The initial stages of this translation involve natural language understanding, visual perception, etc - but these are not specifically mathematical techniques. Later stages involve the specifically mathematical technique of *formula extraction* from the internal meaning representation, which may itself involve *deduction* (see section 3.5).

- Learning: This term covers a multitude of processes, for instance, the learning of new mathematical theorems, the learning of new proof methods, the defining of new concepts, the conjecturing of results, etc. New theorems must be assimilated into the theorem proving ability so that they may be used effectively in the future. The *assimilation* technique required will depend particularly on the search control technique being used, e.g. the new theorem will need to be labelled so that it can be accessed when needed. Similarly, new proof methods must be incorporated into the current search control technique. This may involve a *proof analysis* technique to analyse new proofs and generalize them to extract control information, e.g. proof plans. Defining new concepts can be done by *inductive inference* from descriptions of examples and non-examples of the concept.[2] Conjecturing of theorems can be done by considering the hypothesis of the conjecture to be a concept to be defined using the above techniques.

- Using Analogy: Mathematicians use analogy to suggest conjectures and new definitions, and to guide proofs. All uses require *analogical matching* techniques to find and apply the relationship between the target and the source of the analogy (see [Owen 85] for a survey). This is all that is required for suggesting conjectures and definitions, but to use a source proof to control the search for a target proof a further *proof plan* application technique is required.

This is by no means an exhaustive list; mathematicians also find counterexamples, write, publish and deliver papers, teach students, read textbooks, etc. Unfortunately, we have nothing to say about these other abilities, so they are omitted.

3. Mathematical Reasoning Techniques

In this section we classify and discuss some of the techniques that have been developed for automating the mathematical abilities described in the last section. There is only space to discuss those we consider particularly promising. We group these techniques according to their computational purpose and mutual similarity, e.g. deduction, search control. inductive inference. etc. This provides a more scientific classification of mathematical reasoning processes than the list of mathematical abilities above. But even this classification is bound to be improved as we discover new techniques and gain a better understanding of the existing techniques and their inter-relationships. It should therefore be regarded as both incomplete and highly preliminary.

3.1. Deduction

The most well known deduction technique is *resolution,* [Robinson 65]. Resolution is a rule of inference for Predicate Calculus, that is. it is a rule for deducing new logical formulae from old. To prove that a conjecture is a theorem of a mathematical theory both the axioms of the theory and the negation of the conjecture are expressed as clauses (a normal form for Predicate Calculus formulae). Resolution takes a pair of clauses and makes subparts of each identical using a *matching* technique called *unification.* The remaining parts are combined together to make a new clause. Resolution applied repeatedly to the initial clauses and to their successors defines a search space of clauses. This is searched for a contradiction. If the search is successful then the conjecture is a theorem.

The representation of axioms and conjectures as logical formulae is given in the texts of mathematics and logic. It is in this sense that I claimed that the knowledge representation problem

Note that inductive infaranca it not tha same as mathematical induction (••• •action 3.2 below), which ia a daductive rule of inference.

was solved to a first approximation.

The resolution search space is guaranteed to contain a contradiction if and only if the conjecture is indeed a theorem. The if part is called completeness and the 'only if part is called *soundness.* The snag *is* that if the conjecture *is not* a theorem then the search may never terminate. The number of new clauses generated rises exponentially, or worse, with the length of the proof. If the proof is non-trivial then either the storage capacity of the computer or the patience of the human operator is exhausted before the proof is found. This is an example of the phenomenon called the *combinatorial explosion*.

Various refinements of or alternatives to resolution have been designed and implemented, with the aim of improving on its space and/or time efficiency without losing completeness. One of the most powerful alternative deduction techniques is the Connection Calculus, [Bibel 82]. These improvements reduce the combinatorial explosion, however, but do not conquer it.

There have also been attempts to avoid the combinatorial explosion by proposing radically new deductive techniques. This has sometimes led to techniques which are efficient at proving restricted classes of theorems, e.g. rewrite rules [Huet & Oppen 80]. It has also led to the reinvention of the wheel - some researchers have rejected the logical approach, only to reinvent it together with the attendant problems of incompleteness and/or combinatorial explosion.

Rather than reject logical deduction techniques it is necessary to augment them with heuristic techniques to control the search for a proof, [Hayes 77]. Search control techniques developed have ranged from weak but general purpose ones, e.g. evaluation functions based on the complexity of the formulae, to powerful but special purpose ones, such as those described in the next section. For a survey see [Bledsoe 77].

3.2. Search Control

A number of powerful search control techniques have been developed by careful analysis of the proofs of human mathematicians to extract the underlying control ideas and express them computationally.

The simplest technique, called *heuristic rules,* is to add these control ideas into the mathematical formulae as preconditions for their application. For instance, the LEX program, [Mitchell *et* a/ 81], uses rewrite rules to symbolically integrate algebraic terms, e.g the rule:

$$\int u\ dv \Rightarrow uv - \int v\ du$$

is used to integrate by parts. One class of terms which this rule successfully integrates is those where u is a variable, x, and dv/dx is a constant multiple of a sine or cosine. This search control knowledge is represented in LEX by appending preconditions to the above rule, i.e.

$$u=x \ \&\ dv=n.trig(x)\ dx \rightarrow \int u\ dv \Rightarrow uv - \int v\ du$$

where n is an integer and
trig any trigonometric function

Note that this precondition is too general; trig includes tan, for which the rule does not work. We discuss this problem in section 3.3 below.

Note that these preconditions can get very complicated since the same rule may be used successfully in a number of different circumstances and the precondition must be the disjunct of these circumstances. Note also that the rule now contains a mixture of factual and control knowledge. These complications can cause difficulties, e.g. in the automatic learning of such rules, so some researchers prefer to separate factual and control knowledge.

For instance, the Boyer/Moore Theorem Prover, [Boyer & Moore 79], represents its control knowledge as procedures for manipulating the mathematical formulae. This theorem prover exploits the relationship between recursion and mathematical induction to guide inductive proofs of the properties of Lisp functions. The recursive definitions of the Lisp functions are first used to symbolically evaluate the conjecture to be proved. This may fail because the conjecture contains arbitrary constants and the evaluation process requires lists with internal structure. This failure is used to guide the choice of induction scheme and variable, so that when symbolic evaluation is applied to the induction conclusion just enough structure will be available to enable the induction step to be performed. We will call this technique, *recursion guidance.*

Note that recursion guidance involves an analysis of the conjecture and its failed proof, and the choice of an appropriate proof technique on the basis of this analysis. This analysis uses meta-concepts[4] to describe the conjecture, its proof and the proof methods, e.g. the terms induction hypothesis, induction step, etc in the Boyer/Moore Theorem Prover or trigonometric function in LEX. These meta-concepts must be discovered by study of existing proofs, introspection, etc. It is in this sense that the knowledge representation problem is only solved to a first approximation, since the number of possible meta-concepts is open ended and the choice of appropriate ones determines the success or failure of the technique.

Or an additional rule uaad for aach circumtances.

[4]Tha term meta' ia uaad bacause tha concepts in quaation daacriba the rapraaantation of the problem, i.a. they ara about it rathar than of it.

This process of analysis and guidance using meta-concepts is itself a reasoning process and can be conveniently represented as deduction - but deduction in a meta-theory, rather than the theory itself. Such a use of deduction to guide deduction is made explicit in the PRESS system, [Bundy & Welham 81], for solving equations. The PRESS meta-concepts describe the equation to be solved, e.g. the number of unknowns, their distance apart, the kind of functions involved, etc, and they describe various methods of solving equations or of achieving useful subgoals, e.g. reducing the number of unknowns, moving them closer together, making them occur within identical subterms, etc. Deduction with these meta-concepts induces an implicit, but highly controlled, search for the solution to an equation. We call this technique, *meta-level* inference (see figure 3-1).

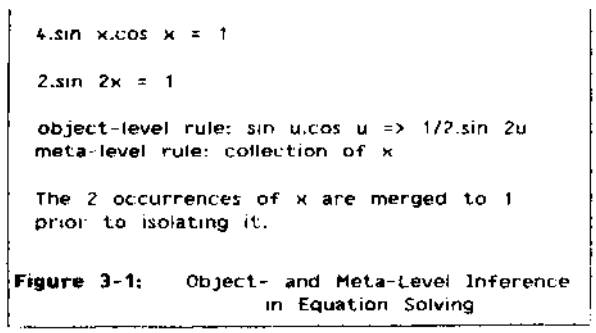

Figure 3-1: Object- and Meta-Level Inference in Equation Solving

These meta-concepts can also be used to express *proof plans,* e.g. the plan to move two unknowns closer together, then merge them and then isolate the remaining occurrence. This plan might be provided by the programmer or learnt by *proof analysis* of a worked example (see section 3.3). Alternatively the proof plan may be at the object-level, e.g. the proof of an analogous theorem, or a generalized proof. [Plummer & Bundy 84]. In each case it must then be applied by a *proof plan application* technique. This application might be straightforward - the target proof exactly following the plan - or the plan may need to be relaxed or augmented at various points. A variety of techniques have been suggested for realising such a flexible plan application technique (see, e.g. [Silver 84])

3.3. Proof Analysis

The process of constructing and augmenting powerful search control techniques, by adding meta-knowledge gained from analysing proofs, can itself be automated. This analysis will depend crucially on the search control technique in question. For instance, in the case of the PRESS system, described in section 3.2. the solution to an equation must be analysed using the meta-concepts of PRESS. For each step an account must be given of not only what algebraic identity was applied but also why it was applied and how this reason fitted into the overall proof plan. For instance, sin u.cos u = 1/2.sin 2u was applied to reduce the occurrences of u from 2 to 1 in order that that single occurrence could be isolated on the left hand side of the equation. This kind of analysis is performed by the LP program. [Silver 84], which was able to learn new methods of equation solving and extend the range of problems that PRESS could solve. The technique used by LP is called *Precondition Analysis* because it discovers new methods by considering how unexplained steps establish the preconditions of successive steps.

In contrast to the deduction of the equation solution, which is called object-level inference.

In the case of the LEX program for symbolic integration (also described in section 3.?) the technique of *back propagation* was developed to analyse successful solutions and extract the control information. This technique was incorporated in the LEX2 program, [Mitchell et *al* 833. In back propagation the successful sequence of rewrite rules was applied in reverse order to a generalized answer to see what constraints this would impose on the original problem. These constraints then became the preconditions of the first rule of the sequence. For instance, given a successful integration of $\cos^7(x)$, LEX? used back propagation to discover that the same sequence of rules would have worked on any term of the form $\cos^n(x)$. where n was an odd integer. The first rule of this sequence,

$$f^r \Rightarrow f^{r-1}.f(x)$$

was then given the precondition, f-cos &, odd(r). In forming this constraint. back propagation also defined the concept. odd* as any number of the form 2k+ 1, where k is an integer.

Proof analysis can also be used to analyse faulty proofs and repair them. For instance, a classic faulty proof in the history of analysis is due to Cauchy. namely that a convergent series of continuous functions is continuous. This proof can be analysed using deduction and the fault identified as a missing occurs check during unification (see section 3.1. This suggests an obvious patch, namely changing the order of quantifiers in the theorem statement, and this generates three new concepts and three new and correct theorems. One of these is the traditional replacement of convergence by uniform convergence; one is a trivial theorem whose conclusion is always true; and one is a new theorem involving the concept of equi-continuity. This reasoning process, which we will call *argument removal,* has been implemented in a program, SEIDEl , described in [Bundy 85].

It would have been more powerful to remember the whole sequence as a proof plan and make the constraints the preconditions of the plan rather than just its first step.

See [Lakatos 76] for a discussion of this proof and its history.

For a survey of the use of proof analysis for learning see [Boswell 85].

3.4. Inductive Inference

Inductive inference has received a lot of attention in the learning literature, with a number of techniques being developed. For a survey see [Dietterich *et al* 82] and for an analytic comparison of some of these techniques see [Bundy et *al* 833. They all learn a concept from examples and, sometimes, non-examples of it. From examples of the concept the techniques might *generalise:* replacing specific relations and terms with more general ones. From non-examples of the concept the techniques might *discriminate:* pruning away relations and terms that are not an essential part of the concept.

This can be used for the learning of both new object-level (factual) and new meta-level (search control). For instance, Mitchell et al's LEX program, [Mitchell *et al* 81], used induction to learn the meta-level conditions for applying a particular strategy for integrating by parts. Given that the strategy worked correctly to integrate 3x.cos(x) and 5x.sin(x), LFX generalized these terms to hypothesise that the strategy will work for terms of the form nx.trig(x), where n is any integer and trig any trigonometric function.

Both the sub-techniques of generalization and discrimination are crucially dependent on the description space. For instance, in LEX the description space contains concepts like 3, 5, n, cos, sin, trig, etc. The generalization of two concepts is the most specific concept in the description space which includes both of them, e.g. trig is the generalization of cos and sin. However, if a term for sine or cosine" were added to the description space then it would replace trig as the generalization of cos and sin. In fact, this is the concept required, as discussed in section 3.1. Without it, LEX overgeneralises to trig.

3.5. Formula Extraction

A key technique in the mathematical formalisation of problems is the extraction of formulae from a meaning representation, e.g. the semantic representation of an English description of the problem. Formulae extraction techniques have received little attention in AI despite their importance in human mathematical reasoning. However, the MECHO program, [Bundy et *al* 79], contained a formula extraction technique called the *Marples algorithm,* for forming equations to describe mechanics problems.

The Marples algorithm is a kind of plan formation technique. Equations are formed by instantiating physical laws, e.g. F=MA. With each law is stored a list of the things it is about and a logical description of how these things relate to the variables in the law. For instance, F=MA is about an object and a direction. The variable M is the mass of the object, A its acceleration in that direction, and F the sum of the forces acting in that direction. A law and a situation are chosen after an analysis of the unknowns and givens of the problem. The variables are then instantiated by inferring the logical description from the meaning of the English problem statement.

This inference process may use deduction and default reasoning to fill gaps between the statement of the problem and the knowledge required to instantiate the law. MECHO used resolution guided by meta-level inference for the deduction and the closed world assumption for the default reasoning. Deduction was needed, for instance, to work out the contributions to the sum of forces from gravity, the tensions of strings, the reactions of contact surfaces, etc. Default reasoning was used, for instance, to assume that the only surfaces in contact were those mentioned in the problem statement.

4. The Interaction of Reasoning Techniques

The mathematical reasoning techniques outlined above can interact in a variety of ways. For instance, successful deductions might provide the material for proof analysis and for analogy. Proof analysis may suggest proof plans to aid deduction. The desire to make a particular theorem hold may trigger a process of proof analysis that leads to changes in definitions and axioms. Sometimes these interactions can be more intimate. Several deduction techniques may be co-routined or even merged so that each prunes the search space of the others. We discuss some of these possibilities in more detail below.

4.1. The Interaction of Techniques in the PRESS Family

In my research group work has continued over a number of years on a family of programs working on the common domain of symbolic equation solving.

- As described in [Bundy &, Welham 81] and section 3.2 above, the PRESS program used the deduction technique of rewrite rules to generate solutions to equations. This deduction technique was guided by the search control technique of meta-level inference.

- The LP (Learning PRESS) program (see [Silver 84] and section 3.3) used the proof analysis technique of precondition analysis to extract and conjecture new methods of solving equations which were then used by PRESS.

- The IMPRESS (Inferring Meta-knowledge about PRESS) program, [Sterling & Bundy 82], was a theorem proving program for proving properties of logic programs. It was used to prove properties of the Prolog code of PRESS using a modified version of recursion guidance (see section 3.2). For instance, it proved the correctness of some PRESS equation solving methods, i.e. that under appropriate preconditions the methods would achieve

their goal.

- As described in [Bundy *et al* 79] and section 3.5 above, the MECHO program solved mechanics problems stated in English by extracting and solving equations using the Marples algorithm. The equation solving part was done using PRESS as a subprogram

Each of the above techniques is incomplete on its own. Meta-level inference requires a rich supply of meta-level concepts to analyse problems and bring to bear appropriate methods of solution. These meta-level concepts can be extracted from example solutions by precondition analysis. The new methods conjectured by precondition analysis can be shown to be correct using recursion guidance. The example solutions required by preconditions can be supplied by deduction with a less constraining search control. The problems to be solved by deduction can be supplied by the Marples algorithm, but this technique requires deduction together with meta-level inference to bridge gaps between the knowledge it requires and that provided in the problem statement. The interactions are summarised in table 4-1.[8]

Technique	Problem	Solution
deduction	search	meta-level inference
	problems	Marples alg.
meta-level inference	learning	pre-condition analysis
pre-condition analysis	verification	deduction/ recursion guidance
	examples	deduction
Marples algorithm	gap bridging	deduction/ meta-level inference

Table 4-1: The Interactions between the Techniques in the PRESS Family

4.2. The Interaction of Learning Techniques

In section 3.4 we described the importance of the description space in constraining the kinds of inductive inference that were possible, old concepts can only be generalized or specialized to new concepts that are contained in the description space. In section 3.3 we described how the techniques of back propagation and argument removal could be used to define new concepts and thus extend the description space in a principled way by adding a needed concept, e.g. odd integer, uniform convergence.

But proof analysis techniques cannot merely replace inductive inference techniques as learning processes. Proof analysis techniques work only on single examples and this limits the amount of generalization that they can do unaided. For instance, we saw that back propagation was able to generalize a particular problem from cos x to $\cos^n x$, where n is an odd integer. 7 is generalized to n by considering the constraints forced by the particular sequence of rules used in the successful solution. However, a similar sequence will also integrate terms of the form $\sin^n x$ where n is an odd integer; a rule for cos needs to be replaced by a similar rule for sin. To recognise this similarity and build a general proof plan for both cases requires inductive generalization, [Boswell 84]. Alternatively, one might use analogical matching to recognise the similarity and a flexible proof plan application technique to apply the cos sequence to the sin problem. Note how back propagation narrows down the search which would otherwise be involved in finding an analogous solution, but then uses analogical matching to further narrow its own search. Therefore, techniques of proof analysis, inductive inference analogical matching and proof application need to work in concert to achieve maximum learning power.

4.3. The Merging of Deduction Techniques

The resolution rule of inference (see section 3.1) is itself formed from the merging of two other rules of inference, namely modus ponens and substitution.

Modus Ponens: A, A->B |- B
where A and B are formulae,

Substitution: A(X) |- A(T)
where A is a formula,
X is a variable and T is a term

Note that the substitution rule of inference has an infinite branching rate if the number of terms in the mathematical theory is infinite. Exhaustive search with such a rule would be totally impractical. In resolution, substitution is made subservient to an upgraded version of modus ponens and its behaviour thereby controlled.

The upgraded version of modus ponens is the cut rule.

Cut: A -> B, B -> C |- A -> C

Faced with formulae A -> B' and B" -> C the resolution rule applies substitutions to either formula in an attempt to make the Bs identical so that cut can be applied. Unification (see section 3.1) will find the most general such substitution. Up to renamings of variables this substitution is unique - a far cry from the infinite branching of undirected substitution. Thus the merging of modus ponens and substitution controls the search implicit in the later rule by making its application subser-

[8] Not all these interactions have been implemented.

vent to the former.

Similar mergings of other deduction techniques have been suggested. For instance, some axioms have been built-in to the unification algorithm. In associative resolution, [Plotkin 72], the associative axiom for a function, f, is deleted from the set of axioms and built-in to the unification algorithm, i.e. it can be used in the attempt to match two expressions. This merging of unification and associativity has the effect of controlling the applications of associativity by making it subservient to unification. Other examples are higher order unification, [Huet 75], which builds the axioms and rules of lambda calculus into the unification algorithm, and E-Resolution, [Morris 69], which builds the equality axioms into the unification algorithm.

The advantage of such mergings is not just in the shrinking of the search space; they can also assist the application of proof plans by bringing the key steps of a proof to the top of the search space. A proof plan may identify a particular step as a key one, but it might take several minor steps to transform the problem into a state where this key step can be applied. For instance, the key step may be to resolve with a particular clause, but several applications of associativity may be required to allow the resolution to go through. These minor steps may create a combinatorial explosion of their own before the key step is reached. By making the minor steps subservient to the key one the key step can be applied first and the application of the minor ones can be controlled.

Merging may also be applicable to non-deduction techniques. For instance, given a problem to solve, an analogical matcher might be able to find a similar solved problem in order to use its solution as a proof plan. Sometimes a solved problem will match the given one in several different ways. Rather than work through each way in turn, the match may be left incomplete until further instantion is required to continue with the proof plan application. Thus the analogical matching will be made subservient to the plan application technique and its search thus controlled.

4.4. Lenat's AM Program

The AM program, [Lenat 8?], is an interesting experiment in the interaction of a number of techniques for finding examples, defining concept, making conjectures, etc. These are set in a framework of heuristic rules controlled by heuristic search. An evaluation function is used to decide what concepts to define or find examples of, what conjectures to make. AM's performance is impressive; starting with some simple set-theoretic concepts, it defines some relatively complex and interesting concepts, e.g. prime numbers, and makes some interesting conjectures, e.g. the prime unique factorization theorem. It also defines a lot of uninteresting concepts and makes a lot of silly conjectures.

AM has no theorem proving ability. Its definitions and conjectures are not motivated by problems it is trying to solve, faulty proofs it is trying to correct or successful solutions it is trying to generalise. Its sense of direction comes entirely from its evaluation function which is guided by the patterns, coincidences, etc that it notices in its example finding. It would be interesting to link AM's techniques to those outlined above to get a better directed process of mathematical discovery. This would involve separating the different techniques used in AM and implementing a wider interaction of mathematical techniques. However, such a programme would not be easy. The techniques used by AM are not clearly explained, are embedded in complicated Lisp code, and are difficult to disentangle from the heuristic rules.

5. The DReaM Programme

In this paper I have advocated the simultaneous study of a number of different techniques for mathematical reasoning, especially how these techniques may be fitted together. I believe that problems associated with the individual reasoning techniques can often be solved by combining them together, and I gave a number of examples of this phenomenon in section 4 above.

To realise these ideas I have instituted the DReaM (Discovery and Reasoning in Mathematics) Programme at the Department of Artificial Intelligence at Edinburgh University. This programme gives explicit recognition to an implicit programme of development of mathematical reasoning programs over a period of several years. Figure 5-1 explains the relationship between the various programs built in our group during this period. Each node is a program and each arc represents some historical dependence of the earlier program upon the later one. PRESS, LP,

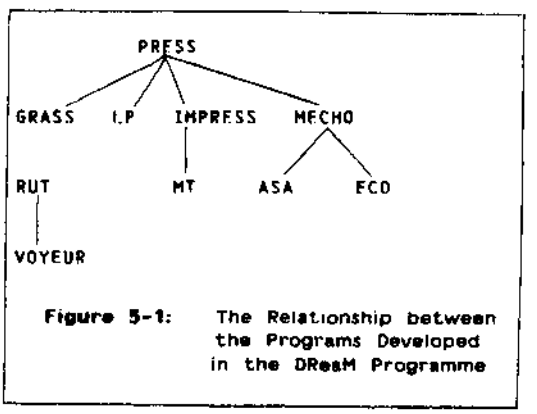

Figure 5-1: The Relationship between the Programs Developed in the DReaM Programme

IMPRESS and MECHO have been described above. MT will be described below. The remainder are outlined here.

- GRASS is a rewrite rule system for Grassman Geometry, modelled on PRESS and using symmetry to control the application of the rules, [Fearnley-Sander 85].
- ECO, [Uschold et al 84], and ASA, [O'Keefe 82], are 'intelligent front ends' to

ecological modelling statistics packages, respectively.

- RUT, [Plummer 84], is a rational reconstruction of Bledsoe's natural deduction theorem prover, [Bledsoe 77], and VOYEUR, [Plummer 8, Bundy 84], extends RUT with the gazing technique described below.

Our specific, short term objectives are to extend our existing reasoning techniques and invent new techniques in a variety of domains, and more centrally to investigate the interaction of: deduction, search control, proof analysis and inductive inference within a single reasoning system. The understanding gained from this investigation will be exhibited in a program for reasoning primarily in mathematics, but adaptable (we hope) to other forms of problem domain.

The core of the system will be the MT program, [Wallen 83], which consists of two parts

- the object-language: a logic for expressing problems; and

  the meta-language: a logic for expressing proof plans.

MT uses a process of meta-level inference to analyse a conjecture, choose an appropriate proof plan and use it to guide the search for a proof.

The object-level deduction technique is based on Bibel's Connection Calculus. [Bibel 82]. The Connection Calculus is particularly suitable as a vehicle for proof plans as it does not demand that the conjecture be put in a normal form and the proof is constructed using a detailed analysrs of the conjecture. No new formulae need to be constructed during the proof. This makes it particularly easy to relate the original analysis of the conjecture to the proof plan and hence to the subsequent proof. We plan to design and implement several proof plans in the MT system. Heuristics developed from natural deduction proofs can be readily translated into the Connection Calculus. In particular, we will try to implement within MT one such technique, developed in the group, called *gazing*, [Plummer & Bundy 84]. Gazing is a heuristic technique for controlling the expansion of non-logical definitions and the use of previously proved theorems during a proof attempt.

We plan to add to MT a learning component based on precondition analysis and other analytic learning techniques. This will analyse proofs using the meta-level concepts already embodied in the MT proof plans and use this analysis to modify the existing plans and/or build new plans. These plans will then be added to MT to improve its theorem proving ability.

6. Conclusion

The automation of mathematical reasoning involves not just techniques for deduction, but also techniques for search control, proof analysis, inductive inference, matching, formula extraction, etc. In this paper I have outlined some of the most promising such techniques drawn from the work of my own group and from that of others.

I have given examples of the interactions of these techniques and shown how these interactions can solve problems which can appear insuperable if a technique is studied in isolation. This constitutes a strong argument for the simultaneous study of reasoning techniques; to see how the total can be more than the sum of the parts.

The DReaM project aims to conduct such a simultaneous study. Some of the preliminary results are reported above together with our plans to incorporate several techniques within a single system.

Acknowledgements

I am grateful to Robin Boswell, Steve Owen, Lincoln Wallen and Mike Uschold for assistance with and/or feedback on this paper.

References

[Bibel 82] Bibel W. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden. 198?.

[Bledsoe 77] Bledsoe, W.W. Non-Resolution theorem-proving. *Artificial Intelligence* 9(1): 1-35, August. 1977.

[Boswell 84] Boswell RA. *Further Developments to an Algebra Learning Program - A Thesis Proposal.* Working Paper 174. Dept. of Artificial Intelligence, Edinburgh, October, 1984.

[Boswell 85] Boswell RA. *An Analytic Survey of Analytic Concept-Learning Programs.* Working Paper. Dept. of Artificial Intelligence, Edinburgh, 1985. Forthcoming.

[Boyer & Moore 79] Boyer, R.S. and Moore J.S. *A Computational Logic.* Academic Press, 1979. ACM monograph series.

[Bundy 83] Bundy, A. *The Computer Modelling of Mathematical Reasoning.* Academic Press. 1983. Earlier version available from Edinburgh as Occasional Paper 24.

[Bundy 85] Bundy. A.
Poof Analysis: A technique for Concept Formation.
In Ross, P. (editor), *Proceedings of AISB-85,* pages 78-86. 1985.
Also available as DAI Research Paper no. 198.

[Bundy & Welham 81]
Bundy, A. and Welham, B.
Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation.
Artificial Intelligence 16(2):189-212, 1981.
Also available as DAI Research Paper 121.

[Bundy et *al* 79]
Bundy, A., Byrd, L., Luger, G., Mellish, C, Milne, R. and Palmer, M.
Solving Mechanics Problems Using Meta-Level Inference.
In Buchanan, B.G. (editor), *Proceedings of IJCAT-79,* pages 1017-1027. International Joint Conference on Artificial Intelligence, 1979.
Reprinted in 'Expert Systems in the microelectronic age' ed. Michie, D., Edinburgh University Press, 1979. Also available from Edinburgh as DAI Research Paper No. 112.

[Bundy et *al* 83]
Bundy, A., Silver, B. and Plummer, D.
An Analytical Comparison of some Rule Learning Programs.
In *Third Annual Technical Conference of the British Computer Society's Expert Systems Specialist Group.* British Computer Society, 1983.
Earlier Version in Procs of ECAI-8?.

[Chang & Lee 73]
Chang C-I. and Lee R. C-T.
Symbolic logic and mechanical theorem proving.
Academic Press, 19 73.

[Diettench *et al* 82]
Dietterich, T.G., London, R., Clarkson, K. and Dromey. G.
Learning and Inductive Inference.
In Cohen, P.R. and Feigenbaum, E.A. (editors), *The Handbook of Artificial Intelligence, Volume 3,* chapter XIV. Pitman Books Ltd, 1982.

[Fearnley-Sander 85]
Fearnley-Sander, D.
Using and Computing Symmetry in Geometry Proofs.
In Ross, P. (editor), *Proceedings of AISB-85.* 1985.

[Hayes 77] Hayes. P.
In defence of logic.
In *Proceedings of IJCAI-77.* International Joint Conference on Artificial Intelligence, 1977.

[Huet 75] Huet, G.
A Unification Algorithm for Lambda calculus.
Theoretical Computer Science 1:27-57, 1975.

[Huet & Oppen 80]
Huet, G. and Oppen, D.C.
Equations and rewrite rules: a survey.
In Book, R. (editor), *Formal languages: perspectives and open problems.* Academic Press, 1980.
Presented at the conference on formal language theory, Santa Barbara, 1979. Available from SRI International as technical report CSI -111.

[Iakatos 76] Lakatos, I.
Proofs and refutations: The logic of Mathematical discovery.
Cambridge University Press, 1976.

[Lenat 82] Lenat D.B.
AM: An Artificial Intelligence approach to discovery in Mathematics as Heuristic Search.
In *Knowledge-based systems in artificial intelligence.* McGraw Hill, 1982.
Also available from Stanford as TechReport AIM 286.

[Mitchell *et al* 81]
Mitchell, T.M., Utgoff. P. E., Nudel, B. and Banerji, R.
Learning problem-solving heuristics through practice.
In *Proceedings of IJCAI-8 1,* pages 127-134. International Joint Conference on Artificial Intelligence, 1981.

[Mitchell *et al* 83]
Mitchell, T.M., Utgoff. P. E. and Banerji, R.
Learning by Experimentation: Acquiring and modifying problem-solving heuristics.
In Michalski, R.S, Carbonell, J.F. and Mitchell. T.M. (editors). *Machine Learning,* pages 163-190. Tioga Press, 1983.

[Morris 69] Morris, J.B.
E-Resolution: Extension of Resolution to include the equality relation.
In Walker, D. and Norton, L.M. (editor), *Proceedings of IJCAI-69,* pages 287-294. Kaufmann Inc., 1969.

[O'Keefe 82] O'Keefe, R.A.
Automated Statistical Analysis.
Working Paper 104, Dept. of Artificial Intelligence, Edinburgh, 1982.

[Owen 85] Owen, S.G.
Analogy in Artificial Intelligence - Thesis Proposal.

Working Paper 176, Dept. of Artificial Intelligence. Fdinburgh. 1985.

[Plotkin 72] Plotkin, G.
Bu»lding-in equational theories.
In Michie, D and Meltzer, B (editors), *Machine Intelligence 7,* pages 73-90. Edinburgh University Press. 1972.

[Plummer 84] Plummer, D.
RUT: Reconstructed UT Theorem Prover.
Working Paper 165, Dept. of Artificial Intelligence, Edinburgh, September. 1984.

[Plummer & Bundy 84]
Plummer, D. and Bundy, A.
Gazing: Identifying potentially useful inferences.
Working Paper 160, Dept. of Artificial Intelligence, Edinburgh, February, 1984.

[Robinson 65] Robinson, J.A.
A machine oriented logic based on the Resolution principle.
J Assoc. Comput. Mach. 12:23-41, 1965.

[Silver 84] Silver, B.
Precondition Analysis: Learning Control Information.
In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (editors), *Machine Learning 2.* Tioga Publishing Company, 1984.
Forthcoming. Earlier version available from Edinburgh as Research Paper 220.

[Sterling & Bundy 82]
Sterling, L. and Bundy, A.
Meta-level Inference and Program Verification.
In Loveland, D.W. (editor), *6th Conference on Automated Deduction,* pages 144-150. Springer- Verlag. 1982.
Lecture Notes in Computer Science No. 138. Also available from Edinburgh as Research Paper 168.

[Uschold *et al* 84]
Uschold, M., Harding, N., Muetzelfeldt, R. and Bundy, A.
An Intelligent Front End for Ecological Modelling.
In O'Shea, T. (editor). *Proceedings of ECAI-84,* pages 761-770. ECAI, 1984.
Available from Edinburgh as Research Paper 223.

[Wallen 83] Wallen, L.A.
Towards the Provision of a Natural Mechanism for Expressing Domain - Specific Global Strategies in General Purpose Theorem-Provers.
Research Paper 202, Dept. of Artificial Intelligence. Edinburgh. September, 1983.